CS304: Automata and Formal Languages

Lec 5

DFA and Regular Languages

Rachit Nimavat

August 7, 2025

Outline

Alphabet Σ: A finite, non-empty set of symbols

String w: A finite sequence of symbols taken from an alphabet

Language L: A set of strings over a particular alphabet

Consider $\Sigma = \{0, 1\}$ and $L = \{0\}^* \{1\} \equiv 0^*1$.

Benefit of Hindsight - C Code

```c
// Returns true if w is in the language 0*1
bool recognize(const char* w) {
    int state = 0; // 0: start, 1: accept, 2: fail
    int len = strlen(w);
    for (int i = 0; i < len; i++) {
        char input = w[i];
        if (state == 0) { // In start state
            if (input == '0') state = 0;
            else if (input == '1') state = 1;
            else state = 2; // Invalid char
        } else if (state == 1) { // accept state
            state = 2; // fail if unexpected char
        }
        // If state is 2 (fail), it stays 2
    }
    return (state == 1);
}
```

- The loop processes the string one character at a time
- The `state` variable is the machine's **only memory**
- The `if/else` logic represents the program's **rules** or **transitions**.

Deterministic Finite Automata (DFA)

Formal Definition

DFA is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$

‘symbol’	description	in our code
Q	finite set of states	$\texttt{state} \in \{0, 1, 2\}$
Σ	underlying finite alphabet	$\Sigma = \{$‘0’, ‘1’$\}$
δ	transition function between states	the ‘core logic’ in C code
$q_0 \in Q$	start state	$\texttt{state} = 0$
$F \subseteq Q$	accepting states	$\texttt{state} = 1$

Deterministic Finite Automata (DFA)

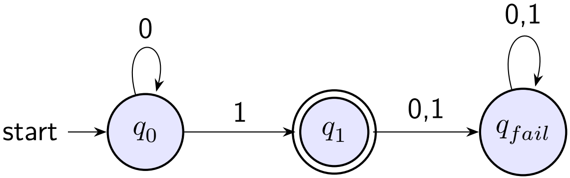
Visualizing DFA

State Diagram is the intuitive way we visualize and work with DFAs

Conventions:

- States Q are circles
- Start state (q_0) has an incoming arrow labeled start
- Accepting states (F) are double circles
- Transitions (δ) are arrows between states, labeled with input symbols from Σ

Our DFA for $L = 0^*1$:



How does a DFA compute?

The 'standard' transition function δ defines a single step of computation.
$\delta : Q \times \Sigma \mapsto Q$ answers: If I am in state q and read symbol a, what state do I move to?

The 'extended' transition function $\hat{\delta}$ defines a leap of computation.
$\hat{\delta} : Q \times \Sigma^* \mapsto Q$ answers:
If I am in state q and read a (sub)string w, what state do I move to?

Recursive Defn of $\hat{\delta}$.

Base case. For any $q \in Q$, define $\hat{\delta}(q, \epsilon) = q$

Recursive step. For any $q \in Q$ and string $w \in \Sigma^*$, define $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$

Where does successively applying δ for each symbol of w leads us to?

e.g. $w = 001$ on DFA $A = (Q, \Sigma, \delta, q_0, \{q_1\})$ for $L = 0^*1$. $\hat{\delta}(q_0, \epsilon) = q_0$, $\hat{\delta}(q_0, 0) = q_0$, $\hat{\delta}(q_0, 00) = q_0$, $\hat{\delta}(q_0, 001) = q_1$.

DFA and Languages

Consider a DFA $A = (Q, \Sigma, \delta, q_0, F)$. A **accepts** a string $w \in \Sigma^*$ if $\hat{\delta}(q_0, w) \in F$.

Language of a DFA. The language recognized by a A, denoted $L(A)$, is the set of all strings that A accepts:

$$L(A) = \left\{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F \right\}$$

Regular Languages. A language L is called a **regular language** if there exists some DFA A such that $L = L(A)$.

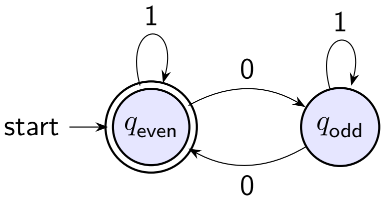
Example 1

$\Sigma = \{0, 1\}$. $L = \{w \mid w$ has even number of $0\}$.
What do we need to remember as we read a string? We only need to know if the count of 0s seen so far is even or odd.

State q_{even}: we've seen even number of 0s (also, the start state)

State q_{odd}: we've seen odd number of 0s

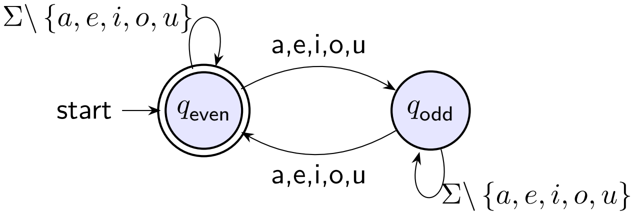
Accepting states? q_{even}



Example 2

$\Sigma = \{a, b, \dots, z\}$. $L = \{w \mid w$ has even number of vowels$\}$.
What do we need to remember as we read a string? We only need to know if the count of vowels seen so far is even or odd.

State q_{even}: we've seen even number of vowels (also, the start state)
State q_{odd}: we've seen odd number of vowels
Accepting states? q_{even}



Example 3

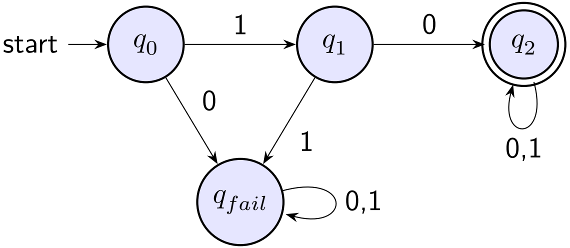
$\Sigma = \{0, 1\}$. $L = \{w \mid w$ starts with $10\}$ Fate is sealed by the first two symbols. We need the following states to track our progress:

q_0: start state. We haven't read any symbols yet.

q_1: first symbol was 1. Hoping for a 0 next.

q_2: first two symbols were 10. Accepting state. Any symbol we see from now on keeps us in this accepting state.

q_{fail}: seen a sequence that makes it impossible to start with 10.



Example 4

$\Sigma = \{0, 1\}$. $L = \{w \mid w$ does not contain substring $11\}$.